`include "config.sv"

module sp\_ram\_wrap

#(

parameter RAM\_SIZE = 32768, // in bytes

parameter ADDR\_WIDTH = $clog2(RAM\_SIZE),

parameter DATA\_WIDTH = 32

)(

// Clock and Reset

input logic clk,

input logic rstn\_i,

input logic en\_i,

input logic [ADDR\_WIDTH-1:0] addr\_i,

input logic [DATA\_WIDTH-1:0] wdata\_i,

output logic [DATA\_WIDTH-1:0] rdata\_o,

input logic we\_i,

input logic [DATA\_WIDTH/8-1:0] be\_i,

input logic bypass\_en\_i

);

`ifdef PULP\_FPGA\_EMUL

xilinx\_mem\_8192x32

sp\_ram\_i

(

.clka ( clk ),

.rsta ( 1'b0 ), // reset is active high

.ena ( en\_i ),

.addra ( addr\_i[ADDR\_WIDTH-1:2] ),

.dina ( wdata\_i ),

.douta ( rdata\_o ),

.wea ( be\_i & {4{we\_i}} )

);

// TODO: we should kill synthesis when the ram size is larger than what we

// have here

`elsif ASIC

// RAM bypass logic

logic [31:0] ram\_out\_int;

// assign rdata\_o = (bypass\_en\_i) ? wdata\_i : ram\_out\_int;

assign rdata\_o = ram\_out\_int;

sp\_ram\_bank

#(

.NUM\_BANKS ( RAM\_SIZE/4096 ),

.BANK\_SIZE ( 1024 )

)

sp\_ram\_bank\_i

(

.clk\_i ( clk ),

.rstn\_i ( rstn\_i ),

.en\_i ( en\_i ),

.addr\_i ( addr\_i ),

.wdata\_i ( wdata\_i ),

.rdata\_o ( ram\_out\_int ),

.we\_i ( (we\_i & ~bypass\_en\_i) ),

.be\_i ( be\_i )

);

`else

sp\_ram

#(

.ADDR\_WIDTH ( ADDR\_WIDTH ),

.DATA\_WIDTH ( DATA\_WIDTH ),

.NUM\_WORDS ( RAM\_SIZE )

)

sp\_ram\_i

(

.clk ( clk ),

.en\_i ( en\_i ),

.addr\_i ( addr\_i ),

.wdata\_i ( wdata\_i ),

.rdata\_o ( rdata\_o ),

.we\_i ( we\_i ),

.be\_i ( be\_i )

);

`endif

endmodule